Automatic Cataloguing and Characterization of Earth Science Data Using SE-trees

Ron Rymon[*]
University of Pennsylvania
rymon@linc.cis.upenn.edu

Nicholas M. Short Jr.[†]
NASA/Goddard Space Flight Center
short@dunloggin.gsfc.nasa.gov

Abstract

In the near future, NASA's Earth Observing System (EOS) platforms will produce enormous amounts of remote sensing image data that will be stored in the EOS Data Information System. For the past several years, the Intelligent Data Management group at Goddard's Information Science and technology Office/930.1 has been researching techniques for automatically cataloguing and characterizing image data (ADCC) from EOS into a distributed database [Cromp *et al.*, 92]. At the core of the approach, scientists will be able to retrieve data based upon the contents of the imagery. The ability to automatically classify imagery is key to the success of contents-based search.

We report results from experiments applying a novel machine learning framework, based on Set-Enumeration (SE) trees [Rymon, 93], to the ADCC domain. Following the design of [Chettri *et al.*, 92], we experiment with two images: one taken from the Blackhills region in South Dakota, the other from the Washington DC area. In a classical machine learning experimentation approach, an image's pixels are randomly partitioned into training (i.e. including ground truth or survey data) and testing sets. The prediction model is built using the pixels in the training set, and its performance estimated using the testing set. With the first Blackhills image, we perform various experiments achieving an accuracy level of 83.2%, compared to 72.7% reported by Chettri *et al.* using a Back Propagation Neural Network (BPNN), and 65.3% using a Gaussian Maximum Likelihood Classifier (GMLC). However, with the Washington DC image, we were only able to achieve 71.4%, compared with 67.7% reported for the BPNN model, and 62.3% for the GMLC.

1 The Problem

In the near future, NASA's Earth Observing System (EOS) platforms will produce enormous amounts of remote sensing image data. An EOS platform will typically carry a number of instruments, each capable of continuously taking measurements on several "channels" of the EM spectrum. Each platform is expected to produce an order of 100 gigabytes of such raw

[*] This work was supported in part by a graduate fellowship ARO Grant DAAL03-89-C0031PRI while the author was in the University of Pennsylvania. Address for correspondence: Ron Rymon, 5600 Munhall Road #207, Pittsburgh, PA 15217.

[†] Address for correspondence: Nicholas Short, Code 930.1, NASA/Goddard Space Flight Center, Greenbelt, MD 20771

C-3

9

data a day; the EOS Data and Information system alone will maintain around 11 petabytes of mostly science image data. A number of previously untackled research problems arise along the various phases of such unprecedented venture: planning observations, transmitting the information to the ground stations, standard product generation and archiving, and finally accessing and analysis by scientists of various disciplines.

Each of these activities can benefit from automated techniques that interpret, analyze, and summarize image data. The Intelligent Data Management (IDM) project of NASA/GSFC/930.1 has developed a paradigm for managing very large databases, called the Intelligent Information Fusion System (IIFS) [Cromp *et al.*, 92]. In addition to providing techniques for retrieval and science data management, the IIFS manages the processing/reprocessing process by first preprocessing image data based on real-time extraction of scientific features for browsing. By providing a glimpse of the contents of the data, scientists can minimize processing/reprocessing rates by prioritizing product generation scheduling. A major part of IIFS, called Automatic Data Characterization and Cataloguing (ADCC), focuses on classification techniques of imagery according to known features (e.g, forestation, urban area, etc) for the generation of browse products and post-retrieval analysis and verification.

While each of these classification tasks is important by itself, and while they share much of their functionality, they represent different tradeoffs between the accuracy of classification and the resources (primarily time) required to attain it. For the purpose of retrieval, real time performance is most important whereas for analysis tasks accuracy is often the top consideration. Algorithms that can be tailored to execute in both the real-time *and* post-retrieval/analysis phase, despite their very distinct accuracy-runtime tradeoffs, can provide a unified framework for simplifying the process.

2 SE-tree-based Classification

About a decade ago, researchers in machine learning (e.g. ID3 [Quinlan, 86]) and statistics (e.g. CART [Breiman *et al.*, 84]) proposed using a *decision tree* as an underlying structure in classification algorithms. A large body of work since then has resulted in significant improvements to algorithms and understanding; thousands of systems and applications followed suit. In the remote sensing domain, for example, [Civco, 89] has shown that decision trees can be built to provide for impressive interpretation accuracy. A new learning framework which generalizes the decision tree-based approach by using a more expressive *Set Enumeration* (SE) tree has recently been presented in [Rymon, 93]. In a recent study, yet to be published, SE-trees were empirically shown to enjoy a particular advantage over decision trees in large domains (for which relatively few examples are available), and in noisy domains. In the remainder of this section we provide a brief and *informal* overview of this approach; the interested reader is referred to [Rymon, 93] for a more in-depth presentation.

2.1 Set Enumeration Trees

SE-trees were first proposed by [Rymon, 92] simply as a way to systematically search a space of sets. As a representation for a decision making procedure, an SE-tree differs from a decision tree in that *multiple* attributes are typically used in each of its branching points. The construction process, and the way in which the SE-tree is used in classification of new instances, are both very similar to the process in which decision trees are commonly constructed and used. In fact, the construction process generalizes the construction of a decision tree, except that here the corresponding partitions are often *not* mutually exclusive. In fact, one way to view an SE-tree is as an edge-sharing *collection* of numerous decision trees. However, taking advantage of their overlap, an SE-tree is not nearly as large as that collection. Similarly, the SE-tree-based classification algorithm generalizes the one which uses decision trees, except that here a *number* of paths can match a new instance.

Given a set of attributes, or features, a *complete* SE-tree is simply a tree representation of all *sets* of attribute-value pairs. It uses an indexing of the underlying set of attributes to do so systematically, i.e. to *uniquely* represent *all* such sets. Figure 1 depicts a complete SE-tree for the space of partial instantiations of three binary attributes (A, B, and C). Assuming alphabetic indexing, notice that a node is only expanded with attributes alphabetically greater than its own (these attributes are referred to as that node's *View*). Of course, the complete SE-tree is typically too large to be used in practice, and so an algorithm will usually search the most relevant parts of that tree.

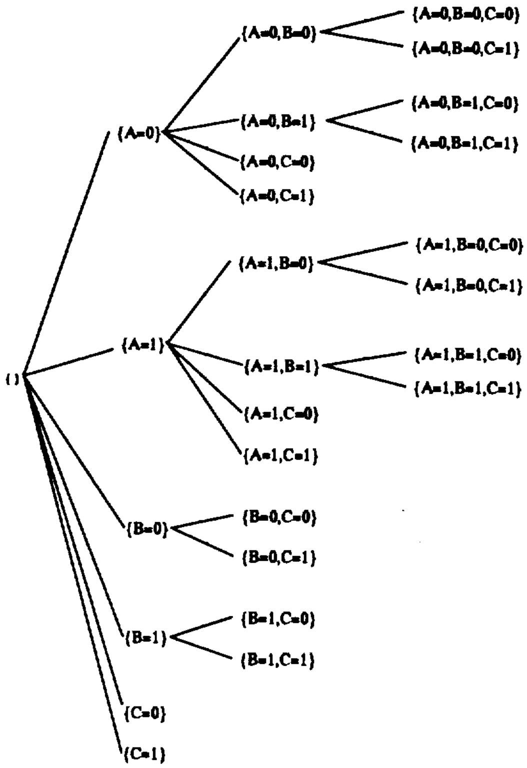


Figure 1: Complete SE-tree for 3 Binary Variables

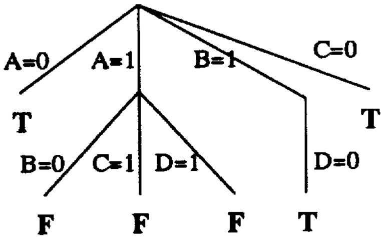
2.2 *SE-Learn* and *SE-Classify*

SE-Learn and *SE-Classify* are, respectively, families of SE-tree-based algorithms for acquiring knowledge from examples (learning), and for using such knowledge to classify new instances. The SE-tree underlies *SE-Learn*'s *search* for classification rules, and is used as an efficient *representation* for these rules, both for the purpose of storing them and for the purpose of subsequently using them in *SE-Classify*.

Like its decision tree counterparts, *SE-Learn* works by recursively partitioning the training set. Unlike decision tree-based frameworks, however, this process stops not only with the discovery of a rule-node, but can also be truncated with the dismissal of a node as not having the potential of leading to rules. Typically, *SE-Learn* starts with all training instances at the SE-tree's root, and first branches on all attribute-value pairs appearing in the training set. According to the SE-tree's structure, it then recurs on a node's remaining training instances, but only with respect to attributes in that node's *View*. *SE-Learn* stops partitioning either when all remaining training instances are equally classified, or by a similar stopping criterion which designates the node's label as a rule (e.g. if there is a great majority for a given class). Rule nodes become leaves; they are retained and labeled with the appropriate class. While searching for rules, *SE-Learn* prunes away nodes which are subsumed by previously discovered rules. Furthermore, its systematic exploration allows it to prune nodes that cannot *lead* to rules, e.g. when the remaining instances only differ in their assignment to attributes *not* in the node's *View*.

Consider, for instance, the following set of instances, and the SE-tree constructed for it by *SE-Learn*:

A	B	C	D	Class
0	0	1	0	T
0	1	1	1	T
1	0	1	0	F
1	1	0	0	T
1	1	1	1	F



Given an SE-tree acquired by *SE-Learn* and a new instance to be classified, the decision procedure *SE-Classify* works by recursively branching from the root of the SE-tree constructed by *SE-Learn* on paths that agree with that instance. Consider the previous example, and a new instance {A=1,B=0,C=0,D=1}. Note that unlike in a decision tree, given its multiple-attribute branching, an SE-tree may include a *number* of such paths. For the new instance, these are the ones labeled with {A=1, B=0}, and {C=0} respectively. *SE-Classify* follows such paths until reaching a class-labeled leaf. Given the potential multiplicity, however, different paths may be labeled with different classes. In our example, for instance, the first rule is labeled with a F whereas the second is labeled with a T. Different variations of *SE-classify* take one of two approaches, or a combination thereof. The first approach involves prioritizing paths (or parts thereof, e.g. some features may be more reliable than others), and consequently the classes labeling their leaves; the algorithm predicts the class labeling the highest priority path. The second approach involves traversing all such paths and using a so-called resolution criteria to determine a prediction from the *combination* of classes. In the remote sensing domain, for example, all rules can be weighted by the reliability or relative relevance of their specific components. Hybrid approaches involve such resolution among the class-labels of a subset of paths with the highest priorities.

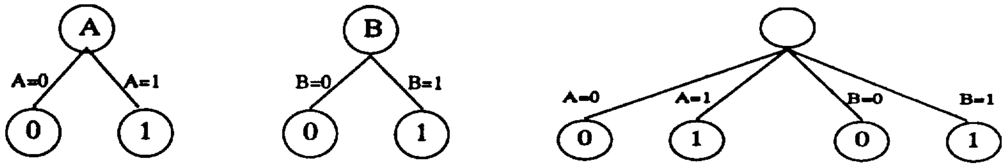
2.3 Discussion

The SE-tree's power as a classification tool draws from the fact that it allows considering a large number of decision procedures in different ways in *each* classification decision. Consider for example the following (rather small) data set and illustration of the two-binary-attribute domain:

A	B	Class
0	0	0
1	1	1

		B	
		0	1
A	0	0	?
	1	?	1

Four different hypotheses are consistent with this training data, corresponding to any assignment of 0 or 1 to {A=0,B=1} and {A=1,B=0}. In contrast, there are only *two* ID3-style[1] decision trees, depicted in Figure 2(a). The corresponding SE-tree (Figure 2(b)) contains, as subset of its arcs, both trees. As we will soon demonstrate, it can be used to represent *all four* hypothesis



(a) Decision Trees

(b) SE-Tree

Figure 2: SE-tree versus Decision Trees

Both *SE-Learn* and *SE-Classify* allow prioritization via user-specified *exploration policies*. Technically, an exploration policy is simply a ranking of sets of attribute-value pairs. In *SE-Learn*, it is used to guide a *best-first* exploration. In *SE-Classify*, it allows implementing *preferences* by prioritizing the paths on which the decision procedure branches.

The role of an exploration policy in *learning* is dual. First, if all minimal rules for the given problem are explored, then the particular order in which nodes are expanded can affect the number of nodes that have to be explored vis-a-vis pruning. In the remote sensing domain, for example, an infra-red channel is known to be a good separator for water-based classification. Thus, relevant problems, branching on such channels first is likely to reduce the tree size. More importantly, if an SE-tree exploration is truncated before all rules are discovered then the exploration policy determines *which* are discovered. The similarity between SE-trees and decision trees allows us to borrow many of the techniques developed for the latter. First, we know that the number of nodes that have to be explored depend on the indexing function used in the SE-tree. A number of techniques developed for attribute selection in decision tree-based frameworks (see, for example, studies by [Mingers, 89a, Buntine & Niblett, 92]) can be used here to select a good indexing function. Second, to reduce exploration, but also to reduce overfitting, we can take advantage of statistically-motivated pruning techniques developed for decision trees (e.g. [Mingers, 89b])

The exploration policy plays an even more important role in *classification*, where it is used to implement *preference* (*bias*). Since training data is often incomplete, a large number of classifiers may be consistent with it, yet differ significantly on new data. The notion of *bias*, the set of preferences guiding the choice among such classifiers has thus received significant attention in the Machine Learning literature (e.g. [Mitchell, 80, Utgoff, 86]). In the remote sensing domain, classification bias can be defined by the particular channel's reliability and/or separability, by the relevance of the feature, and even by the particular data chosen by the scientist for training (since such choice may itself represent some preference). *SE-Classify* allows separating

[1] There are more decision trees, but only these can be generated by an ID3-like procedure.

genuine domain-based preferences from those that merely reflect the particular algorithm used for learning. In many cases, the latter (termed *search bias* [Buntine, 90]) can be eliminated altogether. In variations of *SE-Learn* where exploration is truncated, the exploration policy reflects a learning-phase bias.

Consider again the previous example where we argued that the SE-tree can represent all four hypotheses. Indeed, the particular hypothesis selected in a particular classification session depends on the exploration policy (bias) used. Assuming a classification algorithm which follows arcs by their ranking and predicts the first leaf to be encountered, an OR function[2], for example, can be achieved using an exploration policy which ranks the SE-tree arcs in the following order: B=1 first, A=0 second, A=1 third, and B=0 fourth. The algorithm's response to each possible instance is depicted by the following table:

Instance	Arc followed	Class
{A=0,B=0}	A=0	0
{A=0,B=1}	B=1	1
{A=1,B=0}	A=1	1
{A=1,B=1}	B=1	1

Exploration policies and truncated learning can also be used to trade off accuracy and time, both in learning and in classification. In addition to simple truncation, of particular interest may be a variation of *SE-Learn*, described in [Rymon, 93], that learns *consistent* SE-trees, i.e. ones in which all paths matching a new instance are equally labeled.

Finally, in [Rymon, 93], we show that the SE-tree's indexing function can be dynamically tweaked so that the first nodes to be explored by *SE-Learn* will correspond to those explored by *any given* decision tree-based algorithm. This decision tree will then appear at the leftmost side of the SE-tree. As a result, a variation of *SE-Learn* was implemented which starts off with a given decision tree, and hill-climbs in the performance space. Viewed differently, this result shows once more that a decision tree is a special case of an SE-tree - one in which every node expands with only one new feature and in which every possible instance matches at most one path.

To summarize, the SE-tree can be both viewed and constructed as a generalized decision tree. The similarity between *SE-Learn*/*SE-Classify* and decision tree-based algorithms allows borrowing from the many techniques that were developed for the latter, e.g. discretization algorithms, selection criteria, pruning techniques, etc.

3 Experiments

A prototypical system which implements a variation of *SE-Learn* has been built in the University of Pennsylvania and tested on a number of domains. For the purpose of this paper, we have mainly experimented with Landsat-derived imagery data of South Dakota's Blackhills region. Each pixel in the image corresponds to $79m \times 79m$ on the ground. For every pixel, we have readings from four sensors set to different channel radiances. The readings are each digitized to an integer between 0 and 255. The task is to classify each pixel according to its correct "ground truth", i.e. urban, agricultural, range, forest, or barren. Figure 3 describes the distribution of pixels.

[2] Not modeled by either decision tree.

Class	Name	# pixels	% of total
0	Urban	6676	2.55%
1	Agricultural	42432	16.19%
2	Rangeland	16727	6.38%
3	Forested land	194868	74.34%
6	Barren	1441	0.55%
	Total	262144	

Figure 3: Distribution of Pixels for Blackhills Image

In our experiments, we tried to stay close to the design of [Chettri *et al.*, 92], who had experimented with two classification techniques: Back Propagation Neural Networks (BPNN), and Gaussian Maximum Likelihood Classifier (GMLC). Chettri *et al.* have used the Blackhills and the Washington DC data sets in their experiments. Accuracy results from their experiments, estimated on a holdout from each data set, are presented in the first part of Figure 4. [Campbell *et al.* 89] have also experimented with the Washington DC image, albeit using a more refined geographical classification system.

Experiment description	Accuracy
Back Propagation Neural Network	72.7%
Gaussian Maximum Likelihood Classifier	65.3%
SE-Learn/SE-Classify	
exp. 1 (invq)	47.0%
exp. 1 (vote)	75.5%
exp. 1 (quad)	80.0%
exp. 2 (w/ cluster-based discretization)	83.2%
exp. 3 (w/ four neighbors)	89.8%
exp. 4 (w/ bit-based discretization)	81.6%
exp. 5 (w/ bit disc. & stat-significance pruning)	
ID3 decision tree only	75.9%
SE-tree	78.5%

Figure 4: Comparative Accuracy on Blackhills

In our first experiment with *SE-Learn/SE-Classify*, we have used the Blackhills data set as is, i.e. we associated one attribute with each sensor reading. A fifth attribute was associated with the class, taking 5 values. Learning was pursued to completion, i.e. until all relevant parts of the SE-tree are exposed. A conjunction was defined as a rule if and only if all training instances conforming to it were labeled with the same class. We have experimented with three different *domain independent* weight-based resolution criteria:

1. majority (simple) voting among all matching rules (*vote*);
2. quadratic weighting (*quad*) in which every rule is weighted by r^2 (r denotes cardinality); and
3. inverse quadratic weighting (*invq*) where every rule is weighted by $1/r^2$.

As displayed by Figure 4, there was a great deal of variation in the performance and resolution criteria favoring more specific rules did significantly better.

One problem with the design of the first experiment is that, at least currently, the SE-tree framework handles poorly ordered features. In particular, it is unable to consider *ranges* in such features, and thus has to come up with many more rules than are really necessary. This problem is common to many discrete classification methods. It is common to precede such programs with a process in which all attributes are discretized. Thus, in our second experiment, we added a clustering-based discretization scheme which partitioned the 0-255 range of each sensor reading into a number of intervals. These intervals were then given a name and were input to the *SE-Learn/SE-Classify* program as nominal values. As shown by Figure 4, this has resulted in an improved accuracy. As aside, the difference between the different resolution criteria has almost disappeared, i.e. similar results were obtained using any of the three resolution criteria.

Although the results obtained thus far were already impressive, we tried to milk even more out of the data by allowing the program to look also at a pixel's four neighbors. Given the substantial increase in the size of the training data, we had to add bias in the *learning phase*. In particular, we decided to prefer a pixel's own readings to those of its neighbors. Figure 5 presents accuracy results for ten different tree sizes. We believe that the ultimate improvement in accuracy resulted from the fact that regions tend to be homogeneous, i.e. that a pixel is likely to belong to the same class as its neighbors. The effectiveness of this bias is demonstrated by the temporary *drop* in accuracy, occurring immediately as soon as the algorithm starts considering neighboring pixels. The range of accuracies and their overall correspondence to the tree size (and thus runtime) represents a potential for a dynamic choice of tradeoff. The overall monotonicity suggests that accuracy can be further improved.

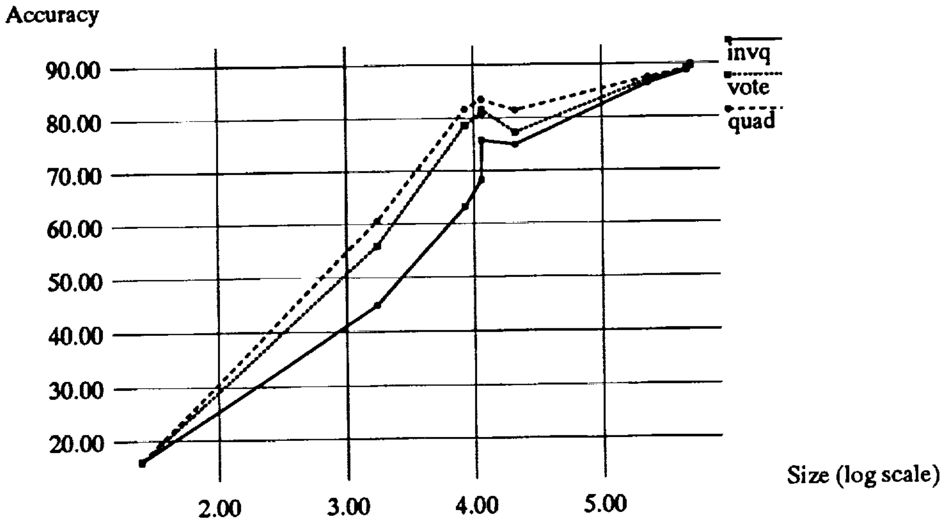


Figure 5: Potential Tradeoff between SE-tree Size and Accuracy

In a fourth experiment, we have discretized the input data by associating a vector of 8 boolean features with each sensor reading, for a total of 32 features per pixel. Again, for lack of time, learning was not pursued to completion. Instead, the program first explored an ID3-based decision tree as the leftmost part of the SE-tree, and then augmented this tree with more parts of the SE-tree in a hill-climbing fashion. However, given the sheer size of the complete SE-tree, we could not run this process to completion. The result corresponds to the last partial SE-tree we could expose, with a simple voting resolution criterion used in classification.

Finally, in a fifth experiment, using the same discretization as above, we added a pruning procedure in which a conjunction is defined as a rule (and is thus not further refined) as soon as one of the classes becomes statistically significant in it (considering the priors of course). In this case, the ID3-based decision tree (pruned as above) was first explored, and new rules were then added by their relative desirability according to Quinlan's information-gain measure. The first result reported corresponds to the decision tree; the second to an SE-tree consisting of this decision tree plus few thousand rules. The pruned decision tree consists of only 7 rules, compared with several thousand rules in the unpruned decision tree.

Having completed the experiments on the Blackhills image, we turned to the Washington DC image. Here, we only ran an experiment similar to the the latter Blackhills experiment, i.e. attributes were bit-discretized, and statistically insignificant rules were pruned away. Figure 6 presents results from this experiment

Experiment description	Accuracy
Back Propagation Neural Network	67.7%
Gaussian Maximum Likelihood Classifier	62.3%
SE-Learn/SE-Classify	
ID3 decision tree only	67.7%
SE-tree	71.4%

Figure 6: Comparative Accuracy on Washington DC

References

[Breiman *et al.*, 84] Breiman, L., Friedman, J., Olshen, R., and Stone, C., *Classification and Regression Trees.* Wadsworth, Belmont, 1984.

[Buntine, 90] Buntine, W., Myths and Legends in Learning Classification Rules. *Proc. AAAI-90*, Boston, MA, pp. 736-742, 1990.

[Buntine & Niblett, 92] Buntine, W., and Niblett, T., A Further Comparison of Splitting Rules for Decision-Tree Induction. *Machine Learning*, 8(1), pp. 75-86, 1992.

[Campbell *et al.* 89] Campbell W. J., Cromp, R. F., and Hill, S. E., Automatic Labeling and Characterization of Objects Using Artificial Neural Networks. *Telematics and Informatics*, 6(3-4):259271, 1989.

[Cromp *et al.*, 92] Cromp, R. F, Campbell, W. J., and Short, N. M., An Intelligent Information Fusion System for Handling the Archiving and Querying of Terabyte-sized Spatial Databases. *Int'l Space Year Conference on Earth and Space Science Information Systems*, Pasadena CA, 1992.

[Chettri *et al.*, 92] Chettri, S. R., Cromp, R. F., and Birmingham M., Design of Neural Networks for Classification of Remotely Sensed Imagery. *Proc. Goddard Conference on Space Applications of Artificial Intelligence*, Greenbelt, MD, 1992.

[Civco, 89] Civco, D., Knowledge-Based Land Use and Land Cover Mapping, *Proc. of 1989 ASPRS/ACSM*, Baltimore, MD, Vol. 3, pp. 276-91.

[Mingers, 89a] Mingers, J., An Empirical Comparison of Selection Measures for Decision Tree Induction. *Machine Learning*, 3(3), pp. 319-342, 1989.

[Mingers, 89b] Mingers, J., An Empirical Comparison of Pruning methods for Decision Tree Induction. *Machine Learning*, 4(2), pp. 227-243, 1989.

[Mitchell, 80] Mitchell, T. M., The Need for Biases in Learning Generalizations. *Technical Report 5-110*, Rutgers University, 1980.

[Quinlan, 86] Quinlan, J. R., Induction of Decision Trees. *Machine Learning*, 1(1), pp. 81-106, 1986.

[Rymon, 92] Rymon, R., Search through Systematic Set Enumeration. *Third International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge MA, pp. 539-550, 1992.

[Rymon, 93] Rymon, R., An SE-tree-based Characterization of the Induction Problem. *Proc. International Conference on Machine Learning*, pp. 268-275, Amherst MA, 1993.

[Utgoff, 86] Utgoff, P. E., *Machine Learning of Inductive Bias.* Kluwer Academic, Boston MA, 1986.